Published before October 2001
Terms used **statement boundary terminator delimiter**

Found **4,239** of **118,149**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Ada, C, C++, and Java vs. the Steelman
David A. Wheeler
July 1997 **ACM SIGAda Ada Letters**, Volume XVII Issue 4
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, citings, index terms

This paper compares four computer programming languages (Ada95, C, C++, and Java) with the requirements of "Steelman", the original 1978 requirements document for the Ada computer programming language. This paper provides a view of the capabilities of each of these languages, and should help those trying to understand their technical similarities, differences, and capabilities.

2 Fortran 8X draft
Loren P. Meissner
December 1989 **ACM SIGPLAN Fortran Forum**, Volume 8 Issue 4
Full text available: pdf(21.36 MB) Additional Information: full citation, abstract, index terms

Standard Programming Language Fortran. This standard specifies the form and establishes the interpretation of programs expressed in the Fortran language. It consists of the specification of the language Fortran. No subsets are specified in this standard. The previous standard, commonly known as "FORTRAN 77", is entirely contained within this standard, known as "Fortran 8x". Therefore, any standard-conforming FORTRAN 77 program is standard conforming under this standard. New features can b ...

3 The equivalence of models of tasking
Daniel M. Berry
January 1972 **Proceedings of ACM conference on Proving assertions about programs**, Volume , 7 Issue 14 , 1
Full text available: pdf(2.12 MB) Additional Information: full citation, abstract, references, citings, index terms

A technique for proving the equivalence of implementations of multi-tasking programming languages is developed and applied to proving the equivalence of the contour model and a multi-tasking version of the copy rule.

4 PDEL—a language for partial differential equations
Alfonso F. Cárdenas, Walter J. Karplus
March 1970 **Communications of the ACM**, Volume 13 Issue 3

Full text available: pdf(880.12 KB) Additional Information: full citation, abstract, references, citings

Conventional computer methods available to solve continuous system problems characterized by partial differential equations are very time-consuming and cumbersome. A convenient, easy to learn and to use, high level problem oriented language to solve and study partial differential equation problems has been designed; a practical translator for the language has also been designed, and a working version of it has been constructed for a significant portion of the language. This P
Keywords: PL/1, PL/1 preprocessor PL/1, finite difference algorithms, partial differential equations, problem oriented or digital simulation language, translator

5 ALGOL Supplement No. 5
Peter Naur
November 1960 **ALGOL Bulletin**, Issue Sup 5
Full text available: pdf(1.58 MB) Additional Information: full citation, index terms

6 GULP—A compiler-compiler for verbal and graphic languages
R. J. Pankhurst
January 1968 **Proceedings of the 1968 23rd ACM national conference**
Full text available: pdf(1.09 MB) Additional Information: full citation, abstract, references, citings, index terms

The General Utility Language Processor (GULP for short) has to operate interactively in a small machine. Its design has to be radically different from compiler-compilers for general-purpose languages in batch-processing operating systems.1,2,3,5 Except for the processing of interrupts, the processing time is unimportant, provided there is a response from the system within a few seconds. Hence every effort was made to reduce storage requirements, even at the expense of p ...

7 Automating the software design process by means of software design and documentation language
Henry Kleine
June 1978 **Proceedings of the 15th conference on Design automation**
Full text available: pdf(766.48 KB) Additional Information: full citation, abstract, references, index terms

The Software Design and Documentation Language (SDDL) approach to software design automation consists of (1) a design and documentation language with forms and syntax that are simple, unrestrictive, and communicative, (2) a processor which can take design specifications written in the SDDL language and produce an intelligible, informative, machine reproducible document, and (3) methodology for effective use of language and processor for top-down design development. The language and processo ...

8 SLANG a problem solving language for continuous-model simulation and optimization
Joe M. Thames
August 1969 **Proceedings of the 1969 24th national conference**
Full text available: pdf(1.00 MB) Additional Information: full citation, abstract, references, citings, index terms

SLANG is a mathematical problem modeling and solution language. It is one of several languages in the programming subsystem of the Computer User Executive (CUE) System developed by TRW Systems. SLANG is both a procedural and a command language designed primarily for the "casual" user. Consequently, much attention was paid to programming ease, "natural" syntax rules, readability, and debugging ease. On the other hand, SLANG is designed to permit the solution of very s ...

9 Continuous system simulation languages: Design principles and implementation techniques

Richard E. Fairley

March 1974 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN symposium on Very high level languages**, Volume 9 Issue 4

Full text available: pdf(626.54 KB) Additional Information: full citation, abstract, references, index terms

Continuous system simulation languages are very high level programming languages which facilitate modelling and simulation of systems characterized by ordinary and partial differential equations. This paper discusses design principles and implementation techniques for continuous system simulation languages. Following a brief introduction to very high level languages, design principles for continuous system simulation languages are presented. These principles are illustrated by ex ...

10 Technical contributions: Some ramifications of the EXIT statement in loop control

M. W. Whitelaw

August 1985 **ACM SIGPLAN Notices**, Volume 20 Issue 8

Full text available: pdf(334.20 KB) Additional Information: full citation, abstract, references

EXIT statements support a style of programming that distinguishes between the control operations and the actions in a loop. Readability suggests that multiple EXIT statements are preferable to a single EXIT statement in a loop with multiple invariants. This is particularly true if implementational considerations have confused the neatness of the loop termination condition.

11 An introduction to ALGOL: a tutorial paper on ALGOL with explanations and examples to make the use of the ALGOL report more familiar

H. R. Schwarz

February 1962 **Communications of the ACM**, Volume 5 Issue 2

Full text available: pdf(1.53 MB) Additional Information: full citation, references, citings

12 Direct Execution In A High-Level Computer Architecture

Yaohan Chu

December 1978 **Proceedings of the 1978 annual conference**

Full text available: pdf(870.65 KB) Additional Information: full citation, abstract, references, index terms

A high-level computer architecture is one where its structure reflects the constructs of high-level programming languages. This paper describes the structure of a high-level computer architecture, which makes use of the basic concepts of control flow and data flow of programming languages. In this structure, there are the lexical, control and data processors to handle the lexical, control and data elements, respectively. Each processor is associated with an associative memory, and the assoc ...

Keywords: Associative memory, Computer architecture, Control processor, Data processor, Direct execution, High-level architecture, Interactive system, Lexical processing

13 A graphical tool for the prototyping of real-time systems

C. J. Coomber, R. E. Childs

April 1990 **ACM SIGSOFT Software Engineering Notes**, Volume 15 Issue 2

Full text available: pdf(813.38 KB) Additional Information: full citation, abstract, citings, index terms

This paper describes a prototyping tool for the design and execution of real-time system

specifications known as transformation schemas. The tool comprises an editor that makes full use of windows, menus, and icons; and a simulator that executes transformation schemas based on an object-oriented strategy. The tool not only enables the syntactic correctness of a transformation schema to be verified, but also assists in proving its semantic correctness.

Keywords: graphical interface, object-oriented programming, prototyping, real-time, simulation, smalltalk, transformation schema

14 Fragmentation and part of speech disambiguation

Jean-Louis Binot

April 1987 **Proceedings of the third conference on European chapter of the Association for Computational Linguistics**

Full text available: pdf(657.05 KB) Publisher Site Additional Information: full citation, abstract, references

That at least some syntax is necessary to support semantic processing is fairly obvious. To know exactly how much syntax is needed, however, and how and when to apply it, is still an open and crucial, albeit old, question. This paper discusses the solutions used in a semantic analyser of French called SABA, developed at the University of Liege, Belgium. Specifically, we shall argue in favor of the usefulness of two syntactic processes: fragmentation, which can be interleaved with semantic proces ...

15 Computer-assisted software testing

Robert V. Fultyn

January 1982 **Proceedings of the ACM '82 conference**

Full text available: pdf(451.00 KB) Additional Information: full citation, abstract, index terms

Obtaining appropriate coverage of functionality while attempting to test software products frequently becomes a troublesome issue to the party responsible for performing the tests. A method that formally specifies the "language" of the software subject being tested and also utilizes some aspects of Monte Carlo simulation techniques for automatic test case selection is described. Metrics useful for attempting to objectively describe quality aspects of the te ...

16 A standard test language - GOAL (Ground Operations Aerospace Language)

Terry R. Mitchell

June 1973 **Proceedings of the 10th workshop on Design automation**

Full text available: pdf(654.45 KB) Additional Information: full citation, abstract, references, index terms

Over the past few years, a team of Kennedy Space Center Civil Service software engineers has specified a standard language for test and ground aerospace operations. The language is intended to be used for test procedure definition for all levels of Space Shuttle flight and ground support hardware from line replaceable units through integrated vehicle testing, and will support both manual and automatic testing modes. That is, the language can not only be used to specify test procedures in th ...

17 A description of the APT language

S. A. Brown, C. E. Drayton, B. Mittman

November 1963 **Communications of the ACM**, Volume 6 Issue 11

Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, citings

The APT (Automatically Programmed Tools) language for numerical control programming is described using the metalinguistic notation introduced in the ALGOL 60 report. Examples of APT usage are included. Presented also are an historical summary of the development of

APT and a statement concerning its present status.

18 FLOWTRACE, a computer program for flowcharting programs

Philip M. Sherman
December 1966 **Communications of the ACM**, Volume 9 Issue 12

Full text available: pdf(1.13 MB) Additional Information: full citation, abstract, references, citings

The FLOWTRACE system produces flowcharts of programs written in "almost any" programming language. One must describe the syntax of the control statements in his language; for this purpose a metalanguage is available. The resultant object deck is used to flowchart any programs in the language described. Several examples of FAP and SNOBOL flowcharts are given. However, it is not necessary to confine one's scope to existing languages. One may define his own language in a ...

19 A parallel compilation technique based on grammar partitioning

Sanjay Khanna, Arif Ghafoor, Amrit Goel
January 1990 **Proceedings of the 1990 ACM annual conference on Cooperation**

Full text available: pdf(873.40 KB) Additional Information: full citation, abstract, references, index terms

A new data partitioning scheme for parallel compilation is presented. The scheme is based on partitioning the grammar, which in turn effectively decomposes the language specification into multiple subsets. The scheme requires definition and development of specialized subcompilers. A simple and practical method is suggested for the implementation of such subcompilers using Parser Generators. The proposed scheme is illustrated through an example, using the grammar of Pascal language. A compar ...

20 A commentary on the ALGOL 60 Revised Report

R. M. De Morgan, I. D. Hill, B. A. Wichmann
December 1974 **ALGOL Bulletin**, Issue 38

Full text available: pdf(1.29 MB) Additional Information: full citation, index terms

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next